# Adaptive Strategy Selection for Concept Learning

William M. Spears and Diana F. Gordon
Naval Research Laboratory
Washington, D.C. 20375

## Abstract

In this paper, we explore the use of genetic algorithms (GAs) to construct a system called GABIL that continually learns and refines concept classification rules from its interaction with the environment. The performance of this system is compared with that of two other concept learners (NEWGEM and C4.5) on a suite of target concepts. From this comparison, we identify strategies responsible for the success of these concept learners. We then implement a subset of these strategies within GABIL to produce a multistrategy concept learner. Finally, this multistrategy concept learner is further enhanced by allowing the GAs to adaptively select the appropriate strategies.

**Key words:** concept learning, genetic algorithms

## 1 Introduction

An important requirement for both natural and artificial organisms is the ability to acquire concept classification rules from interactions with their environment. In this paper, we explore the use of an adaptive search technique, namely, genetic algorithms (GAs), as the central mechanism for building a system, called GABIL, that continually learns and refines concept classification rules from its interaction with the environment. We show how concept learning tasks can be represented and solved by GAs, by explaining GABIL in some detail.

We then provide empirical results that compare the performance of GABIL with two other concept learning systems (NEWGEM and C4.5) on a suite of artificially designed target concepts that increase in complexity, as well as one natural target concept. From this comparison, we identify the mechanisms, called *strategies*, that we consider to be largely responsible for each system's superiority on certain classes of target concepts.

In this paper, we consider a strategy to be the combination of a generalization operator and the criterion for firing this operator. What we consider to be a strategy has the effect of setting a concept learning parameter. For example, using a strict criterion with a generalization operator will cause that operator to fire infrequently or not at all. Likewise, using a liberal criterion will cause it to fire often. The choice of generalization operators and the frequency and conditions under which they fire will have a global effect on the type of concept learning that is done. Therefore, we can consider these strategies to be *learning preferences* (see (Michalski, 1983)), which are also called *inductive biases*.

Concept learning preferences have traditionally been implemented using a generate-and-test approach (i.e., generate hypotheses then select those that meet the preference criteria). Our approach consists of strategies that examine the criteria (test) and *then* fire the operator (generate). This test-and-generate approach, which is similar to that of *test incorporation* (Braudaway and Tong, 1989), can be considerably more efficient than generate-and-test.

Although the level of granularity of generalization operators is finer than that of strategies such as induction and analogy, the issues at *any* level of granularity are remarkably similar. For example, which strategies will enhance/degrade system performance? Will multiple strategies interfere?

To address these issues, we implement two strategies within GABIL. We then present experiments that test the combination of these strategies. It is shown that although these strategies enhance system performance, subtle effects can sometimes cause them to interfere with each other. Furthermore, the best strategy for learning one concept may not be the best for another concept. Therefore, it is difficult to choose optimal combinations of strategies prior to the concept learning task. In response, we have modified GABIL to adaptively shift between strategies when appropriate.

Adaptive strategy selection, in our context, is similar to dynamic preference (bias) adjustment for concept learning (see (Gordon, 1990) for related literature). GABIL is the first system to perform dynamic preference adjustments for concept learning using genetic algorithms, which are naturally suited to adaptive tasks. GAs are also well suited to searching multiple dimensions in parallel, a quality that is needed when seeking the best mixture of strategies, where each strategy may be considered a separate dimension.

## 2 GAs and Concept Learning

Supervised concept learning involves inducing descriptions (i.e., inductive hypotheses) for the concepts to be learned from a set of positive and negative *examples* of the target concepts. An example is an instance that is classified as positive or negative. Examples are represented as points in an $n$-dimensional feature space which is defined a priori and for which all the legal values of the features are known. Concepts are therefore represented as subsets of points in the given $n$-dimensional space. A concept learning program is presented with both a description of the feature space and a set of correctly classified examples of the concepts, and is expected to generate a reasonably accurate description of the (unknown) concepts.

In order to apply GAs to a concept learning problem, we need to select an internal representation of the space to be searched and define an external function that assigns a fitness to candidate solutions. Both components are critical to the successful application of the GAs to the problem of interest. [1]

[1] Excellent introductions to GAs can be found in (Holland, 1975) and (Goldberg, 1989).

## 2.1 Representing the search space

The traditional internal representation of GAs involves using fixed-length (generally binary) strings to represent points in the space to be searched. However, such representations do not appear well-suited for representing the space of concept descriptions that are generally symbolic in nature, that have both syntactic and semantic constraints, and that can be of widely varying length and complexity.

There are two general approaches one might take to resolve this issue. The first involves changing the fundamental GA operators (crossover and mutation) to work effectively with complex non-string objects (Rendell, 1985). This must be done carefully in order to preserve the properties that make the GAs effective adaptive search procedures (see (DeJong, 1987) for a more detailed discussion). Alternatively, one can attempt to construct a string representation that minimizes any changes to the GAs. In this paper, we only discuss the latter approach.

## 2.2 Fixed-length classification rules

Our approach to choosing a representation that results in minimal changes to the standard GA operators involves carefully selecting the concept description language. A natural way to express complex concepts is as a disjunctive set of (possibly overlapping) classification rules, i.e., in disjunctive normal form (DNF). The left-hand side of each rule (disjunct) consists of a conjunction of one or more tests involving feature values. The right-hand side of a rule indicates the concept (classification) to be assigned to the examples that match its left-hand side. Collectively, a set of such rules can be thought of as representing the (unknown) concepts if the rules correctly classify the elements of the feature space. This collective set of classification rules is the system's inductive hypothesis.

If we allow arbitrarily complex terms in the conjunctive left-hand side of such rules, we will have a very powerful description language that will be difficult to represent as strings. However, by restricting the complexity of the elements of the conjunctions, we are able to use a string representation and standard GAs, with the only negative side effect that more rules may be required to express the concept. This is achieved by restricting each element of a conjunction to be a test of the form:

```
return true if the value of feature i
  of the example is in the given value set;
return false otherwise.
```

For example, a rule might take the following symbolic form:

```
if [(F1 = large) & (F2 = sphere v cube)]
then it is a widget.
```

Since the left-hand sides are conjunctive forms with internal disjunction (e.g., the disjunction within feature F2), there is no loss of generality by requiring that there be at most one test for each feature (on the left hand side of a rule). The result is a modified DNF that allows internal disjunction. (See (Michalski, 1983) for a discussion of internal disjunction.)

With these restrictions we can now construct a fixed-length internal representation for classification rules. Each fixed-length rule will have $N$ feature tests, one for each feature. Each feature test will be represented by a fixed-length binary string, the length of which will depend on the type of feature (nominal, ordered, etc.). Currently, GABIL only uses features with nominal values. The system uses $k$ bits for the $k$ values of a nominal feature. So, for example, if the legal values for feature F1 are the sizes {small, medium, large}, then the pattern 011 would represent the test for F1 being medium or large.

Further suppose that feature F2 has the values {sphere, cube, brick}. Then, as an example, the left-hand side of a rule for a 2 feature problem would be represented internally as:

```
F1  F2
111 100
```

This rule is equivalent to:

```
if [(F1 = small v medium v large) &
   (F2 = sphere)]
then it is a widget.
```

Notice that a feature test involving all 1s matches any value of a feature and is equivalent to "dropping" that conjunctive term (i.e., the feature is irrelevant). So, in the above example only the values of F2 are relevant. For completeness, we allow patterns of all 0s which match nothing. This means that any rule containing such a pattern will not match (cover) any points in the feature space. While rules of this form are of no use in the final concept description, they are quite useful as storage areas for GAs when evolving and testing sets of rules.

The right-hand side of a rule is simply the class (concept) to which the example belongs. This means that our "classifier system" is a "stimulus-response" system with no message passing.

## 2.3 Sets of classification rules

Since a concept description will consist of one or more classification rules, we still need to specify how GAs will be used to evolve sets of rules. There are currently two basic strategies: the Michigan approach exemplified by Holland's classifier system (Holland, 1986), and the Pittsburgh approach exemplified by Smith's LS-1 system (Smith, 1983). Systems using the Michigan approach maintain a population of *individual rules* that compete with each other for space and priority in the population. In contrast, systems using the Pittsburgh approach maintain a population of *variable-length rule sets* that compete with each other with respect to performance on the domain task.

Very little is currently known concerning the relative merits of the two approaches. In this paper we report on results obtained from using the Pittsburgh approach.[2] That is, each individual in the population is a variable-length string representing an unordered set of fixed-length rules (disjuncts). The number of rules in a particular individual is unrestricted and can range from 1 to a very large number depending on evolutionary pressures.

Consider the following example of a rule set with 2 disjuncts:

```
F1  F2  F1  F2
100 100 011 010
```

This rule set is equivalent to:

[2] Previous GA concept learners have used the Michigan approach. See (Wilson, 1987) and (Booker, 1989) for details.

if [(F1 = small) & (F2 = sphere)] v
[(F1 = medium v large) & (F2 = cube)]
then it is a widget.

## 2.4 Crossover and mutation

Our goal was to achieve a representation that requires minimal changes to the fundamental genetic operators (crossover and mutation). Genetic operators modify individuals within a population. Crossover takes two individuals and produces two new individuals, by swapping portions of genetic material (e.g., bits). Mutation simply flips random bits within the population, with a small probability (e.g., one bit per 1000).

We feel we have minimized changes to crossover and mutation with the variable-length string representation involving fixed-length rules. Crossover can occur anywhere (i.e., both on rule boundaries and within rules). The only requirement is that the corresponding crossover points on the two parents "match up semantically". That is, if one parent is being cut on a rule boundary, then the other parent must be also cut on a rule boundary. Similarly, if one parent is being cut at a point 5 bits to the right of a rule boundary, then the other parent must be cut in a similar spot (i.e., 5 bits to the right of some rule boundary). For example consider the following two rule sets:

F1 F2 F1 F2
**10|0 10|0 011 010**
F1 F2 F1 F2
01|0 001 110 01|1

We use a "|" to denote a crossover cut point. Note that the left cut point is offset 2 bits from the rule boundary, while the right cut point is offset 1 bit from the rule boundary. The bits within the cut points are swapped, resulting in a rule set of 3 rules and a rule set of one rule:

F1 F2 F1 F2 F1 F2
**100** 001 110 01**0** **011** **010**
F1 F2
01**0** **101**

It is important to note that crossover performs the task of creating rule sets of varying length. The mutation operator is unaffected and performs the usual bit-level mutations.

## 2.5 Choosing a fitness function

In addition to selecting a good representation, it is important to define a good fitness function that rewards the right kinds of individuals. For the experiments reported in this paper, we selected a fitness function involving only classification performance (ignoring, for example, length and complexity biases). The fitness of each individual rule set is computed by testing the rule set on the current set of examples (which is typically a subset of all the examples - see Section 2.6) and letting:

$$fitness(individual\ i) = percent\ correct$$

This provides a bias toward correctly classifying all the examples while providing differential reward for imperfect rule sets. This bias is equivalent to one that encourages *consistency* and *completeness* of the hypotheses with the examples. A hypothesis is consistent when it covers no negative examples and is complete when it covers all positive examples.

## 2.6 The GA concept learner

Figure 1 provides a pseudo-code description of the genetic algorithm within GABIL:

```
procedure GA;
begin
        t = 0;
        initialize population P(t);
        fitness P(t);
        until (done)
                t = t + 1;
                select P(t) from P(t-1);
                crossover P(t);
                mutate P(t);
                fitness P(t);
end.
```

Figure 1. The GA for GABIL.

P(t) represents a population of rule sets. After a random initialization of the population, each rule set is evaluated with the fitness function (described above). Those rule sets that have higher fitness (i.e., are more consistent and complete) are selected for survival. At this time crossover and mutation are applied to each surviving rule set, to produce a new population. This cycle continues until a consistent and complete rule set has been found.

A standard GA can evolve concept descriptions in a couple of ways: *batch* mode, where all instances are presented to the system at once, and *incremental* mode, where one or a few of the instances are presented to the system at a time. The simplest approach involves using batch mode, in which a fixed set of examples is presented and the GA must search the space of variable-length strings described above for a set of rules that achieves a fitness of 100%. If such a rule set is not found within a user specified length of time, the rule set with the highest fitness is selected (so, in fact, we weaken the consistency and completeness requirements). We call this approach GABL (GA Batch concept Learner).

Since the incremental mode is more adaptable to realistic changing environments, this is the mode we would prefer to use for GABIL. The simplest way to produce an incremental GA concept learner is to use GABL as follows. The concept learner initially accepts a single example from a pool of examples. GABL creates a 100% correct rule set (or as close to 100% as possible) for this example. This rule set predicts the classification of the next example. If the prediction is incorrect, GABL is invoked (in batch) to evolve a new rule set using the two examples. If the prediction is correct, the example is simply stored with the previous example and the rule set remains unchanged. As each new additional instance is accepted, a prediction is made, and the GA is re-run in batch if the prediction is incorrect. We refer to this mode of operation as *batch-incremental* and we refer to the GA batch-incremental concept learner as GABIL. Although batch-incremental mode is more costly to run than batch, it provides a much more finely-grained, and therefore better, measure of performance. Rather than measure an algorithm's performance over a small subset of the instances, batch-incremental mode measures the performance of this algorithm over *all* instances.

## 3 Empirical System Comparisons

The experiments described in this section are designed to compare the predictive performance of GABIL and two other concept learners (NEWGEM and C4.5) as a function of incremental increases in the size and complexity of the target concept.

### 3.1 The domains

We have two domains: one artificial, and one natural. For Domain 1, we invented a 4 feature world in which each feature has 4 possible distinct values (i.e., there are 256 instances in this world).

Within Domain 1, we constructed a set of 12 target concepts. We varied the complexity of the 12 target concepts by increasing both the number of rules (disjuncts) and the number of relevant features per rule (conjuncts) required to correctly describe the concepts. The number of disjuncts ranged from 1 to 4, while the number of conjuncts ranged from 1 to 3. Each target concept is labeled as *n*D*m*C, where *n* is the number of disjuncts and *m* is the number of conjuncts.

For each of the target concepts, a set of 256 unique, noise free examples was generated from the feature space and labeled as positive or negative examples of the target concept. For the more complex concepts, this resulted in learning primarily from negative examples. For each concept, the 256 examples were randomly shuffled and then presented sequentially in batch-incremental mode. This procedure was repeated 10 times (trials) for each concept and learning algorithm pair.

For Domain 2, we used a well-known natural database designed for diagnosing breast cancer (Michalski et. al., 1986). This database has descriptions of cases for 286 patients, and each case (instance) is described in terms of 9 features. There is a small amount of noise in the database. Furthermore, the target concept is considerably more complex than any of the concepts in the *n*D*m*C world. For example, after seeing all 286 instances, the NEWGEM system (described below) develops an inductive hypothesis having 25 disjuncts and an average of 4 conjuncts per disjunct. Since GABIL can only handle nominals, and the breast cancer instances have features in the form of numeric intervals, we converted the breast cancer (BC) database to use nominal features. We have run experiments using NEWGEM and C4.5 and have found that this conversion affects the performance of both systems in roughly the same way (e.g., prediction accuracy increases between 1 and 3%). When using the BC database, we again randomly shuffled the instances and averaged over 10 runs.

### 3.2 The systems

For the artificial domain, GABIL has a population of 1000. For the BC target concept, the population is 100. In addition to GABIL, two well-known concept learners had their performance evaluated on the *n*D*m*C and BC target concepts: NEWGEM (Mozetic, 1985), which is based on the AQ algorithm described in (Michalski, 1983) and is also called AQ14, and C4.5 (Quinlan, unpublished). The C4.5 system is based on the ID algorithm described in (Quinlan, 1986). All systems are run in batch-incremental mode.

NEWGEM, like AQ, generates classification rules from instances using a beam search. NEWGEM maintains two sets of classification rules: one set, which we call the *positive inductive hypothesis*, is for learning the target concept and the other set, which we call the *negative inductive hypothesis* is for learning the negation of the target concept. (Recall that GABIL uses only a positive inductive hypothesis.) NEWGEM, like GABIL, generates classification rules in a modified DNF that allows internal disjunction of feature values. Internal disjunction implies fewer external disjuncts in the hypotheses.

NEWGEM guarantees that its inductive hypotheses will be consistent and complete with respect to all examples. This system's performance depends on its parameter settings. The particular parameter settings that we chose for NEWGEM implement a preference for simpler inductive hypotheses, e.g., inductive hypotheses having shorter disjuncts.[3]

C4.5 uses a decision tree representation rather than a rule representation for its inductive hypotheses. Each decision tree node is associated with an instance feature. The node represents a test on the value of the feature. Arcs emanating from a feature node correspond to values of that feature. Each leaf node is associated with a classification (e.g., positive or negative if one concept is being learned). To view a decision tree as a positive DNF hypotheses, one would consider this hypothesis to be the disjunction of all paths (a conjunction of feature values) from the root to a positive leaf. The negative hypothesis is similar.

An information theoretic measure biases the search through the space of decision trees by using entropy minimization as a criterion for ordering the tree nodes. The result of this information theoretic measure is a preference for simpler (i.e., shorter) decision trees. C4.5 does not require completeness or consistency.

[3] The precise criteria used are: the positive and negative inductive hypotheses are allowed to intersect provided the intersection covers no instances, noisy examples are considered positive, the maximum beam width is set to 20, and the minimum number of features and values are preferred in each disjunct. Other settings, which have less impact on system performance are the system defaults.

## 3.3 Performance criteria

To compare the performance of our batch-incremental mode systems, we have generated learning curves. Each curve represents an average performance over 10 independent trials for learning a single target concept. For each trial, we examine a small window of recent outcomes, counting the correct predictions within that window. The value of the curve at each time step therefore represents the percent correct achieved over the most recent window of instances. The window size was 10 for the artificial domain and 50 for the BC domain (to eliminate excessive peaks in the learning curves).

After generating learning curves for each target concept, we collapsed the information from these curves into four performance criteria. The first, called the *prediction accuracy* (PA) criterion, is the average over all values on a learning curve, from the beginning to the end of learning a target concept. This criterion takes advantage of the fact that we use batch-incremental rather than batch mode because there are many values over which to take an average. The second, called the *convergence* (C) criterion (a la PAC convergence as in (Valiant, 1984)), is the number of instances seen before a 95% prediction accuracy is maintained. It is possible, of course, that 95% prediction accuracy may never be achieved (e.g., on the BC database). The finely-grained measure obtainable with batch-incremental mode facilitates this performance criterion as well.

The criteria just described are considered *local* because they apply to a single target concept. For each local criterion there is a corresponding *global* criterion that considers all target concepts in a domain. The global prediction accuracy criterion is the average of the PA criteria values on every target concept within a domain. Likewise, the global convergence criterion is the average of the C criteria values on all the target concepts of a domain. Since the global criteria are based on far more data than the local criteria, we base our conclusions from the experiments on the former.

### 3.4 Results

Table 1 shows the results of the PA and global PA (denoted ''Average'' in the tables) criteria for measuring the performance of all systems on the *n*D*m*C and BC target concepts, while Table 2 shows the results of applying the C and global C (denoted ''Average'' in the tables) criteria to measure performance on the *n*D*m*C concepts only (since no system achieves 95% prediction accuracy on the BC database). Although there are subtle differences between Tables 1 and 2, the general trend is similar if you group together results that are close. From these tables we can see that NEWGEM (abbreviated ''GEM'' in the tables) is the best performer overall. In particular, NEWGEM is the top or close to the top performer on the *n*D*m*C concepts. This system does not, however, perform as well as other systems on the BC target concept. These results are due to the fact that NEWGEM, when using our chosen parameter settings, is a system that is well tuned to simpler DNF target concepts.

C4.5 performs well on all but the target concepts that have many short disjuncts.[4] GABIL appears to be a good overall performer. It does not do superbly on any particular concept, but it also does not have a distinct region of the space of concepts on which it clearly degrades. Furthermore, GABIL is quite competitive on the difficult BC target concept. Finally, it is important to observe that there is no system that is clearly superior to all others on all the target concepts. Therefore, multistrategy concept learning is the appropriate direction to take.

| Prediction Accuracy | | | |
|---|---|---|---|
| TC | GEM | C4.5 | GABIL |
| 1D1C | 99.78 | 98.50 | 95.24 |
| 1D2C | 98.41 | 96.09 | 95.77 |
| 1D3C | 97.43 | 98.48 | 95.73 |
| 2D1C | 98.64 | 93.41 | 92.00 |
| 2D2C | 96.81 | 94.27 | 92.67 |
| 2D3C | 96.68 | 96.89 | 94.55 |
| 3D1C | 97.98 | 78.77 | 90.40 |
| 3D2C | 95.48 | 92.18 | 90.33 |
| 3D3C | 95.30 | 95.38 | 92.80 |
| 4D1C | 95.77 | 66.41 | 89.55 |
| 4D2C | 93.81 | 90.53 | 87.40 |
| 4D3C | 93.49 | 93.82 | 88.85 |
| Average | 96.63 | 91.23 | 92.11 |
| BC | 60.52 | 72.36 | 68.65 |

Table 1. Prediction accuracy.

## 4 A Multistrategy Concept Learner

After obtaining comparisons of all three systems on target concepts of growing complexity, we decided to use this information to construct a multistrategy concept learner. There are three methods we could use to build such a system: (1) implement a high-level monitor that selects the most appropriate concept learning system (e.g., C4.5 or NEWGEM) for learning each concept, (2) implement a meta-level parameter adjuster that selects parameters (e.g., NEWGEM's *lexicographic*

[4] An explanation of the difficulty of systems based on ID3 on target concepts of this type is in (De Jong and Spears, 1991).

*evaluation function* which contains various system parameters) to optimize a particular system for each target concept, or (3) implement strategies from each of the systems and embed them within one concept learner to make a multistrategy concept learner.

We chose the last of these options. It is important for the reader to note that if we use strategies equivalent to parameters of a single system, then the second and third options have equivalent effects. In other words, implementing a strategy in a system has the effect of setting the equivalent (e.g., lexicographic evaluation function) parameter. To implement the third option, we selected a set of strategies from each concept learner that we considered to be largely responsible for that system's success on a particular class of target concepts. We then implemented some of these strategies within GABIL to make this system a multistrategy learner designed for robust general performance (because GABIL is its basis), as well as for optimized performance on the classes of target concepts for which each of its new strategies (each adopted from a system other than GABIL) is best suited.

| Convergence | | | |
|---|---|---|---|
| TC | GEM | C4.5 | GABIL |
| 1D1C | 13 | 37 | 87 |
| 1D2C | 28 | 155 | 100 |
| 1D3C | 57 | 0 | 96 |
| 2D1C | 28 | 100 | 109 |
| 2D2C | 43 | 126 | 148 |
| 2D3C | 86 | 181 | 249 |
| 3D1C | 34 | 253 | 103 |
| 3D2C | 78 | 122 | 125 |
| 3D3C | 195 | 253 | 225 |
| 4D1C | 82 | 253 | 131 |
| 4D2C | 78 | 113 | 142 |
| 4D3C | 154 | 253 | 229 |
| Average | 73 | 154 | 145 |

Table 2. Convergence to 95%.

Since NEWGEM seemed to be the best overall performer, we began by adding two strategies from this system to GABIL. As we have described above, the NEWGEM used in our experiments has preferences for shorter disjuncts. After studying the NEWGEM system, we hypothesized that this is one of the strategies largely responsible for NEWGEM's superior performance on the *n*D*m*C concepts. We began by implementing NEWGEM's simplicity preference strategy.

## 4.1 Dropping condition strategy

NEWGEM implements its simplicity preference by selecting simpler inductive hypotheses from among those that have already been generated. However, we decided to implement a less costly version of this strategy. We selected the (only) generalization operator of NEWGEM that encourages the initial formation of inductive hypotheses with short disjuncts and implemented this operator within GABIL. By adding a criterion for firing this operator in GABIL, we have converted it into a strategy.

This strategy, which we call the *dropping condition strategy*, drops a feature (i.e., condition) from a disjunct if it seems to be nearly irrelevant within that disjunct. The operator of this dropping condition strategy is based on the generalization operator *dropping condition* described in (Michalski, 1983). This operator drops a feature from a disjunct. For example, if the disjunct is

[(F1 = small v medium) & (F2 = sphere)]

then dropping condition might create the new disjunct

[(F2 = sphere)].

The criterion for this strategy, which is based on a criterion from (Gordon, 1990), examines the bits of each feature in each disjunct. If most (i.e., more than half) of the bits of a feature in a disjunct are 1s, then the remaining 0 bits are changed to 1s. By changing the feature to have all 1 values, this strategy forces the feature to become irrelevant within that disjunct and thereby simulates the effect of a shortened disjunct. To illustrate, suppose GABIL decides to modify the following disjunct:

```
F1  F2
110 100
```

Then the dropping condition operator will result in a new disjunct as follows:

```
F1  F2
111 100
```

Note that feature F1 is now irrelevant within this disjunct. We call GABIL with this dropping condition strategy ''GABIL+D''.

## 4.2 Adding alternative strategy

We added another strategy from NEWGEM, which we also considered to be largely responsible for NEWGEM's superior performance on the artificial target concepts. The purpose of this second strategy is to increase the generality of the inductive hypotheses. This strategy, which we call the *adding alternative strategy*, uses an operator that is based on the *adding alternative* operator of (Michalski, 1983). This operator generalizes by adding a disjunct (i.e., alternative) to the current classification rule. The most useful form of this operator, according to (Michalski, 1983) is when an internal disjunct is added. For example, if the disjunct is

[(F1 = small) & (F2 = sphere)]

then the adding alternative operator might create the new disjunct

[(F1 = small) & (F2 = sphere v cube)].

The *adding alternative strategy* is implemented in GABIL with an asymmetric mutation rate. In particular, there is a 75% probability of mutating a bit to a 1 but a 25% probability of mutating it to a 0. Therefore, the adding alternative operator in GABIL has a strong preference for mutating bits to 1s, i.e., adding internal disjuncts. The criterion for firing the adding alternative operator is the product of the probability (0.01) of mutating a bit and the probability (0.75) of mutating a bit to a 1. To illustrate, the adding alternative strategy might change

```
F1  F2
100 100
```

to

```
F1  F2
100 110
```

Note that feature F2 has been generalized in this disjunct. The addition of the adding alternative to strategy to GABIL results in a system called ''GABIL+A''. When both strategies are added to GABIL, the resulting system is called ''GABIL+AD''.

In a standard genetic algorithm, recall that crossover and mutation are used as genetic operators to produce new offspring from parents. In GABIL, both the dropping condition and adding alternative strategies are implemented in a similar fashion, acting as genetic operators that are used to create new rule sets (i.e., offspring). The two strategies are applied to each rule set, after crossover and mutation, and before the fitness of each individual is evaluated (see Figure 1).

## 4.3 Results

We have run experiments comparing GABIL+A, GABIL+D, and GABIL+AD. Table 3 shows the results of system performance measured using the PA and global PA criteria. Table 4 shows the results of system performance measured using the C and global C criteria. GABIL is abbreviated ''G'' in the tables.

According to the global criteria in Tables 3 and 4, GABIL+A does not perform as well as GABIL+D or GABIL+AD. On the BC target concept, the combination of both strategies (GABIL+AD) is the best. It is interesting to note, however, that on the *n*D*m*C domain, GABIL+AD does not perform as well as the single strategy GABIL+D.

Tables 3 and 4 provide answers to the questions that were posed in the introduction of this paper. There, we questioned which strategies help/hinder learning and whether their combination results in interference. We define interference to be the situation where the performance of the system with multiple strategies is worse than that of the system with one of the individual strategies.

| Prediction Accuracy | | | |
|---|---|---|---|
| TC | G+A | G+D | G+AD |
| 1D1C | 96.14 | 97.65 | 97.65 |
| 1D2C | 96.20 | 97.35 | 97.27 |
| 1D3C | 95.69 | 96.71 | 96.67 |
| 2D1C | 93.09 | 97.38 | 96.95 |
| 2D2C | 95.04 | 96.32 | 96.85 |
| 2D3C | 94.54 | 95.82 | 94.97 |
| 3D1C | 91.89 | 96.04 | 96.63 |
| 3D2C | 91.56 | 94.45 | 94.61 |
| 3D3C | 92.74 | 94.19 | 92.86 |
| 4D1C | 90.90 | 95.08 | 95.22 |
| 4D2C | 89.69 | 92.95 | 92.74 |
| 4D3C | 89.15 | 92.30 | 89.97 |
| Average | 93.06 | 95.52 | 95.20 |
| BC | 69.14 | 71.49 | 72.01 |

Table 3. Prediction accuracy.

Based on the global criteria in Tables 3 and 4, the answer to the first question is that both dropping condition and adding alternative appear to improve performance individually, and dropping condition seems to be the more effective of the two. The answer to the second question is that, on the *n*D*m*C domain, the two strategies in GABIL+AD can interfere with each other since GABIL+AD performs worse than GABIL+D in terms of both global criteria. However, further experiments with a different GA parameter setting (i.e., a population of 100) indicate that this is not always the case. For that parameter setting, GABIL+AD outperforms GABIL+D in terms of both global criteria. These results are consistent with the BC results, which were obtained with a population of 100.

These results indicate that it is not possible to know beforehand which set of strategies is best. Other system parameters may influence the usefulness of each strategy, and how it interacts with others. To address this issue, we modified GABIL to adaptively shift between strategies when appropriate. The following section explains these modifications, and presents the results.

## 5 An Adaptive Concept Learner

Although it would be possible to construct alternative adaptive mechanisms for selecting appropriate strategies, it is rather more natural to allow the GA to perform this search, as well as the search for good hypotheses. A clear advantage of this approach is that no new search mechanism need be written (since the GAs are designed for adaptive search). We next consider how adaptive selection of strategies is implemented in GABIL.

| Convergence | | | |
|---|---|---|---|
| TC | G+A | G+D | G+AD |
| 1D1C | 58 | 28 | 32 |
| 1D2C | 85 | 59 | 68 |
| 1D3C | 97 | 94 | 97 |
| 2D1C | 90 | 42 | 42 |
| 2D2C | 93 | 82 | 55 |
| 2D3C | 250 | 136 | 250 |
| 3D1C | 104 | 54 | 39 |
| 3D2C | 127 | 76 | 62 |
| 3D3C | 240 | 161 | 240 |
| 4D1C | 120 | 67 | 62 |
| 4D2C | 133 | 75 | 75 |
| 4D3C | 253 | 166 | 248 |
| Average | 138 | 87 | 106 |

Table 4. Convergence to 95%.

Recall that each individual of the GA population represents a variable length hypothesis. In the previous sections, the adding alternative and dropping condition strategies are allowed to operate on any given individual. Suppose, however, that we allow each individual to determine which strategies are permissible. In that case, those individuals that enable the selection of the best strategies will survive in the GA, thus performing the search for the best set of strategies and the search for the best hypotheses in parallel.

For the case to be considered here, recall that we are interested in controlling two strategies. Suppose that each individual (hypothesis) has two added control bits. The first bit determines whether the dropping condition strategy can be used on that individual. The second bit determines whether the adding alternative strategy can be used on that individual. Thus, if the control bit is 1, the associated strategy is permissible. If the control bit is 0, the associated strategy is not permissible. These control bits act as added preconditions for the strategies. Thus the GA must select whether it is better to have no strategy, either strategy, or both. In general, one can have *N* bits, to control *N* strategies.

GABIL was modified to include two extra control bits for every individual within the population. The first bit controls the dropping condition strategy, and the second bit controls the adding alternative strategy. For example, consider the following rule set:

```
F1  F2   F1  F2  D A
010 001  110 011 1  0
```

The two added control bits are indicated with the letters "D" and "A" (for dropping condition and adding alternative, respectively).

For this rule set the dropping condition strategy is permissible, while the adding alternative strategy is not.

The GA is allowed to search the space of strategies and the space of hypotheses in parallel (see also (Back et. al., 1991) for related work in GAs). The resulting adaptive system, which we call ''adaptive GABIL'', was run on the *n*D*m*C and BC target concepts. The results are presented in Table 5.

The results of the global criteria, shown at the bottom of Table 5, highlight a couple of important points. First, on the *n*D*m*C domain, the adaptive GABIL outperforms the original GABIL, GABIL+A, and GABIL+AD. Furthermore, the adaptive GABIL performs almost as well as GABIL+D from a prediction accuracy criterion, and better from a convergence

| | PA | C |
|---|---|---|
| TC | | |
| 1D1C | 97.57 | 34 |
| 1D2C | 97.35 | 58 |
| 1D3C | 96.50 | 97 |
| 2D1C | 96.14 | 50 |
| 2D2C | 96.20 | 80 |
| 2D3C | 95.44 | 120 |
| 3D1C | 95.88 | 53 |
| 3D2C | 93.96 | 70 |
| 3D3C | 94.65 | 128 |
| 4D1C | 95.82 | 55 |
| 4D2C | 92.82 | 80 |
| 4D3C | 92.13 | 130 |
| Average | 95.37 | 80 |
| BC | 70.30 | N/A |

Table 5. Adaptive GABIL performance.

criterion. Adaptive GABIL shows a reduction in interference over GABIL+AD, particularly from the standpoint of the global C criterion. It is in this case that we see the virtues of adaptive GABIL, which dynamically selects the appropriate strategies.

On the BC target concept, adaptive GABIL performs better than GABIL and GABIL+A, but is worse than GABIL+D and GABIL+AD. Again, this indicates that GABIL's advantage with a smaller population size is not as significant. To address this issue, future versions of GABIL will have to adapt the population size, as well as strategy selection.

In comparison to the other systems, the new adaptive GABIL is much better than C4.5 on the *n*D*m*C domain, and close on the BC target concept. Also, adaptive GABIL is competitive with NEWGEM on the *n*D*m*C domain, and is much better on the BC target concept. We have tested the statistical significance of these results, and found that when adaptive GABIL outperforms other systems, the results are generally significant (at a 90% level). Furthermore, when other systems outperform adaptive GABIL, the results are generally not significant (i.e., significance is 80% or lower). The only two notable exceptions are on the BC database. Both C4.5 and GABIL+AD outperform GABIL at a 95% level of significance. We believe that the latter exception is due to the small population size. The former exception will be addressed when we incorporate C4.5's strategies into GABIL.

Considering that GABIL is now performing the additional task of selecting appropriate strategies, these results are very encouraging. The implication of adaptive strategy selection is that we now not only have a method for *dynamically* adjusting system parameters (because of the equivalence between system parameter adjustment and strategy selection), but we furthermore have a method for embedding parameters from *multiple* systems within a single system and dynamically adjusting them in a favorable way.

## 6 Discussion and Future Work

The experiments in this paper highlight that no one set of strategies (or concept learning system) is best for all target concepts. We have implemented and compared the performance of a learner that uses an adaptive strategy selection mechanism. Initial results indicate that this is a promising approach for multistrategy learning. Using this approach, we can not only simulate dynamic parameter adjustment for a single system, but furthermore can simulate dynamic adjustment of useful parameters from multiple systems.

So far, we have implemented two strategies from a single system (NEWGEM). In the future, we plan to implement more strategies from that system, as well as strategies from other systems. For example, we would like to implement in GABIL an information theoretic measure, which we believe is primarily responsible for C4.5's successes. This strategy could be implemented by making features with higher entropy values more likely to have 1s.

We have been addressing the issue of strategies at a finer level of granularity, namely, strategies that are generalization operators along with criteria for firing these operators. We believe that the adaptive approach to multistrategy learning will be a promising approach in general. Future research will test this by applying the adaptive method to higher level strategies such as induction and analogy. Our final goal is to produce a multistrategy learner that dynamically adapts to changing concepts and nonstationary learning conditions, which are frequently encountered in realistic environments.

## Acknowledgements

We would especially like to thank Ken De Jong, who made many major contributions to the GABIL project. We would also like to thank the Machine Learning Group at NRL for their useful comments about GABIL, J. R. Quinlan for C4.5, and Zianping Zhang and Ryszard Michalski for NEWGEM.

## References

Back, T., F. Hoffmeister, and H. Schwefel, A Survey of Evolution Strategies, *Proc. 4th Int'l Conference on Genetic Algorithms and their Applications*, 1991.

Booker, L., Triggered Rule Discovery in Classifier Systems, *Proc. 3rd Int'l Conference on Genetic Algorithms and their Applications*, 1989.

Braudaway, W. and C. Tong, Automated Synthesis of Constrained Generators, *Proc. of the Sixth International Workshop on Machine Learning*, 1989.

De Jong, K, Using Genetic Algorithms to Search Program Spaces, *Proc. 2nd Int'l Conference on Genetic Algorithms and their Applications*, 1987.

De Jong, K. and W. Spears, Learning Concept Classification Rules Using Genetic

Algorithms, *Proc. International Joint Conference on Artificial Intelligence*, 1991.

Goldberg, D., *Genetic Algorithms in Search, Optimization & Machine Learning,* Addison-Wesley Publishing Company, Inc., 1989.

Gordon, D., *Active Bias Adjustment for Incremental, Supervised Concept Learning.* Ph.D. thesis, University of Maryland, College Park, MD., 1990.

Holland, J., *Adaptation in Natural and Artificial Systems,* The University of Michigan Press, 1975.

Holland, J., Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. In R. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Vol. 2). Morgan Kaufmann Publishers, Los Altos, CA., 1986.

Michalski, R., A Theory and Methodology of Inductive Learning. In R. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Vol. 1). Tioga Publishing Co., Palo Alto, CA., 1983.

Michalski, R., I. Mozetic, J. Hong, and Lavrac, N., The AQ15 inductive learning system: An overview and experiments. University of Illinois Technical Report Number UIUCDCS-R-86-1260, 1986.

Mozetic, I., NEWGEM: Program for learning from examples, program documentation and user's guide. University of Illinois Report Number UIUCDCS-F-85-949, 1985.

Quinlan, J., Induction of Decision Trees. *Machine Learning,* Volume 1, Number 1, 1986.

Quinlan, J., Documentation and User's Guide for C4.5. (unpublished), 1989.

Rendell, L., Genetic Plans and the Probabilistic Learning System: Synthesis and Results. *Proc. 1st Int'l Conference on Genetic Algorithms and their Applications*. 1985.

Smith, S., Flexible Learning of Problem Solving Heuristics Through Adaptive Search, *Proc. 8th IJCAI*, 1983.

Wilson, S., Quasi-Darwinian Learning in a Classifier System, *Proc. of the 4th Int'l Workshop on Machine Learning*, Morgan Kaufman Publishing, 1987

Valiant, L., A Theory of the Learnable. *Communications of the ACM,* 27, 1984.